aiven

Why should Database people care about Kafka and Debezium?

Dirk Krautschick

Swiss PGDay 27.06.2024

aiven

Why should PostgreSQL people care about Kafka and Debezium?

Dirk Krautschick

Swiss PGDay 27.06.2024

#whoami

Dirk Krautschick
Solution Architect

with Aiven since Nov 2023

16 years

DBA, Trainer, Consulting, Sales Engineering

PostgreSQL, Oracle

Married, 2 Junior DBAs

Mountainbike, swimming, movies, music, hifi/home cinema, 8 bit computing

Disclaimer

Inspiration and motivation for trying

I am NOT an deep dive streaming guru …

… just a desperate database guy …like you? :-)

Unexceptional Open Source!!!

Everything what looks like advertising won’t be meant like such…

Disclaimer

Inspiration and motivation for trying

I am NOT an deep dive streaming guru …

… just a desperate database guy …like you? :-)

Unexceptional Open Source!!!

Everything what looks like advertising won’t be meant like such…

 ***…but still it’s highly recommended!!! ;-)*** 

Motivation… once upon a time…

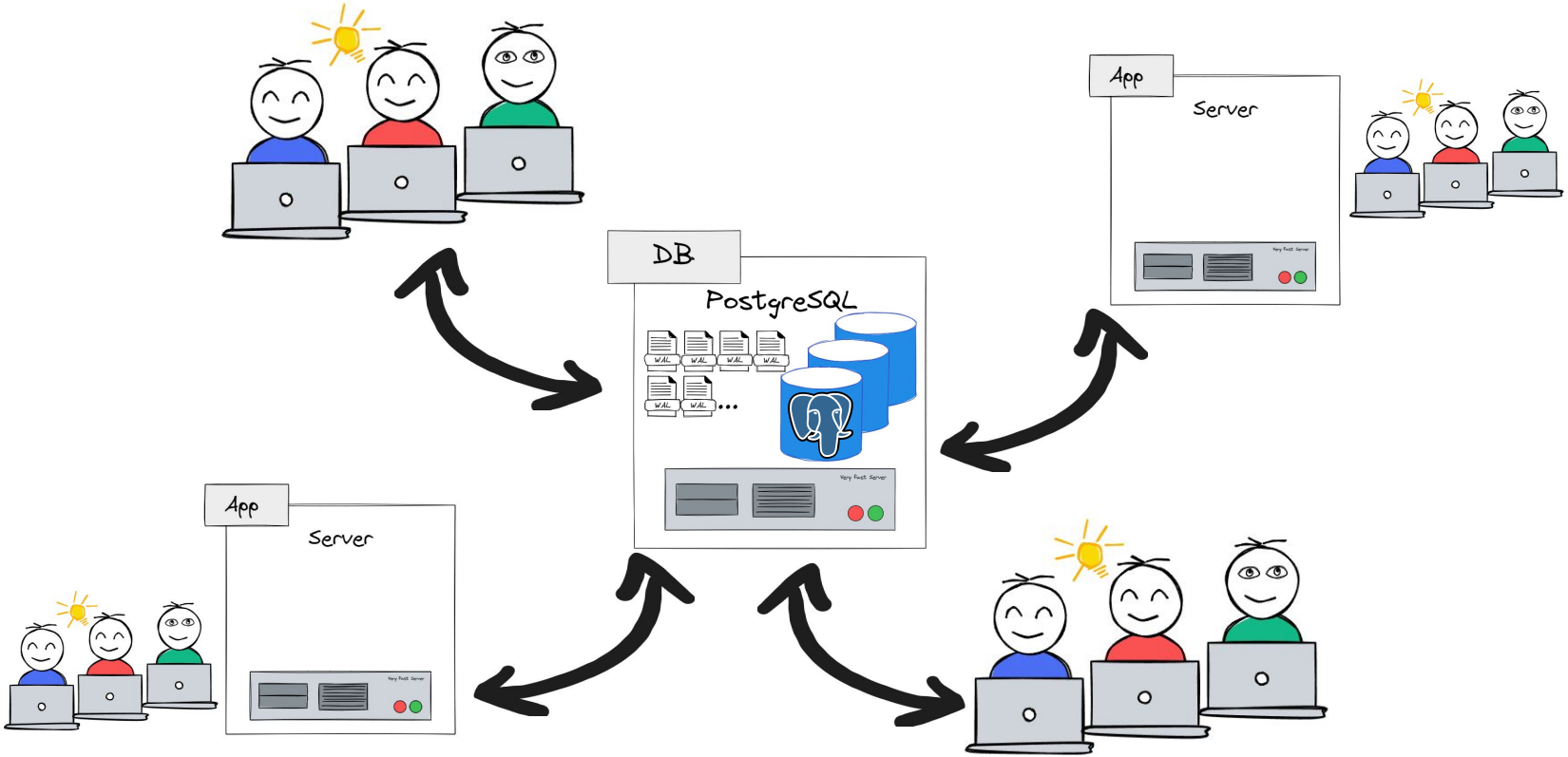
…being a DBA was like….

my server,

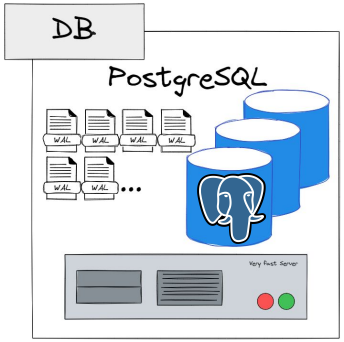
my databases,

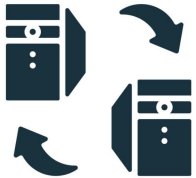
my responsibilities!!!

Motivation… once upon a time…



Motivation… but today…





DATA MIGRATION

Motivation… but today…

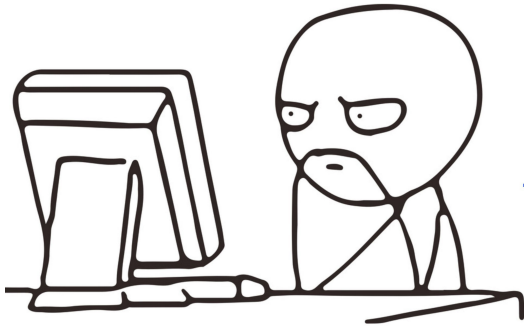
Not just only ordinary clients/applications anymore

Role of DBAs changing

- Cloud/DBaaS vs. on premises
- Architectural and scaling challenges
- Different database types for different use cases

Real time data processing and heterogeneity

Rethink your capabilities!? … like I had to?!



No worries, we'll stick with Databases

Motivation… PostgreSQL rules for all!!!

Extensibility is powerful, thats for sure!

- Foreign Data Wrappers
- Special data types
- …

Some smart interfaces

Just direct connections

...but then...

Idea of a better solution

Decoupled communication/interfaces

Real time processing

Prevent dedicated connections for each use case

Queuing

Change data capture capabilities

Idea of a better solution

Decoupled communication/interfaces

Real time processing

Prevent dedicated connections for each use case

Queuing

Change data capture capabilities

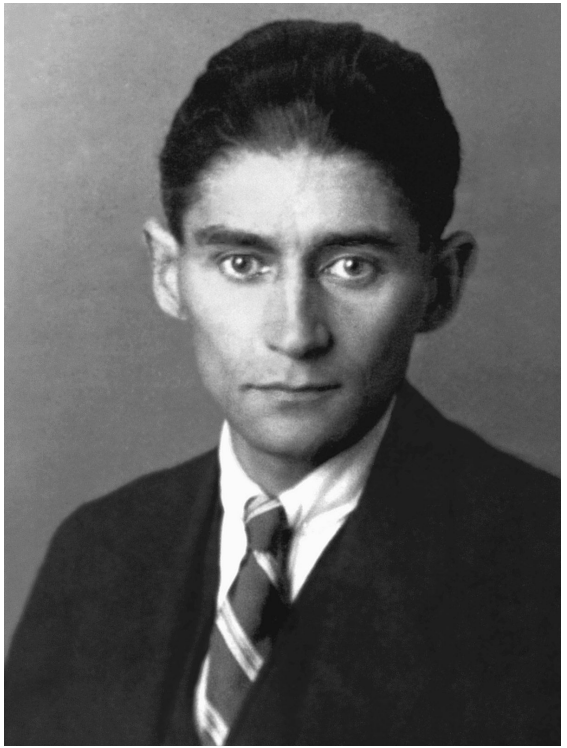
“AT SCALE !!!111eleven” ...of course :-)

→ “Distributes event streaming”

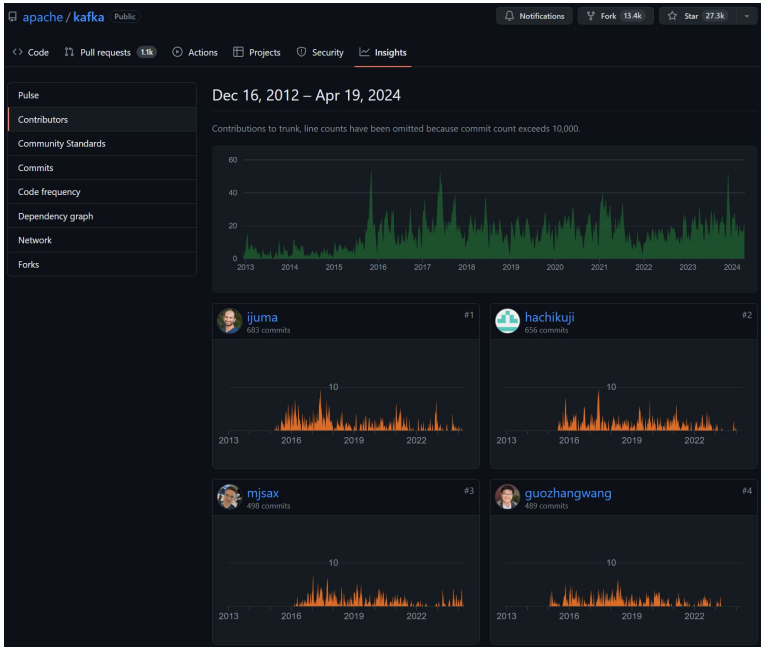
Let’s introduce… Kafka

Franz Kafka (3 July 1883 – 3 June 1924) was a German-speaking Bohemian Jewish novelist and writer from Prague……

….just kidding!



Let’s introduce… Apache Kafka



Apache Kafka

Initiated by LinkedIn (Nov 2010)

Apache since 2012

Apache License 2.0

https://github.com/apache/kafka

Release v1.0.0 (Nov 2017)

Actual Release v3.7.0 (Feb 2024)

Java- and Scala-based

Let’s introduce… Apache Kafka

Apache Kafka

Initiated by LinkedIn **(Nov 2010)**

Apache **since 2012**

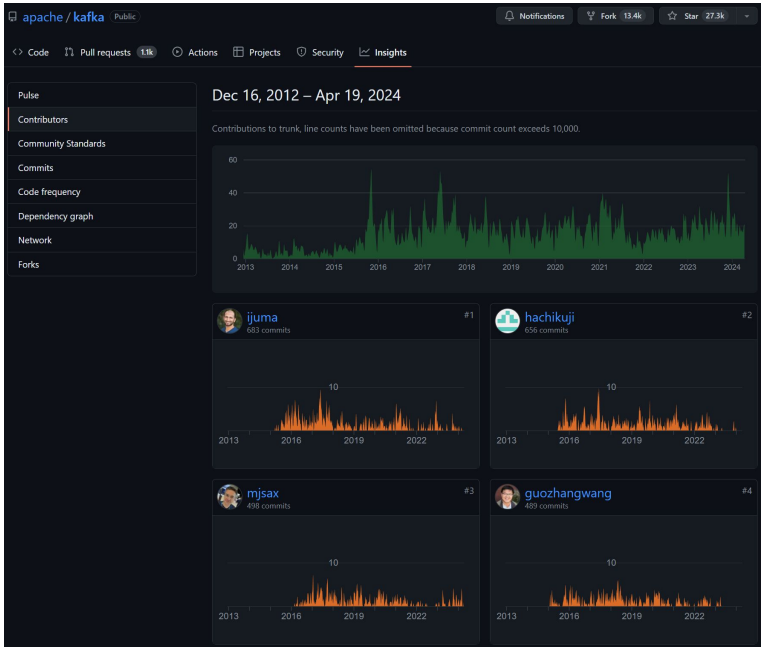
Apache License 2.0

https://github.com/apache/kafka

Release v1.0.0 (Nov 2017)

Actual Release v3.7.0 (Feb 2024)

Java- and Scala-based



Kafka basics

Events vs. Messages vs. Queues

Streaming vs. Oracle Streams vs. Postgres Streaming Replication vs. ....

Be careful with the terms!

Here we are talking

"streaming" of "events", which is usually

Set of key/value pairs

Timestamp, and maybe optional metadata

Kafka basics

Topics

- Organized collection of stored events
- Basically folders/files on storage

Partitions

- Distribution of topic parts on several brokers

Kafka components

Java JRE/JDK (8+)

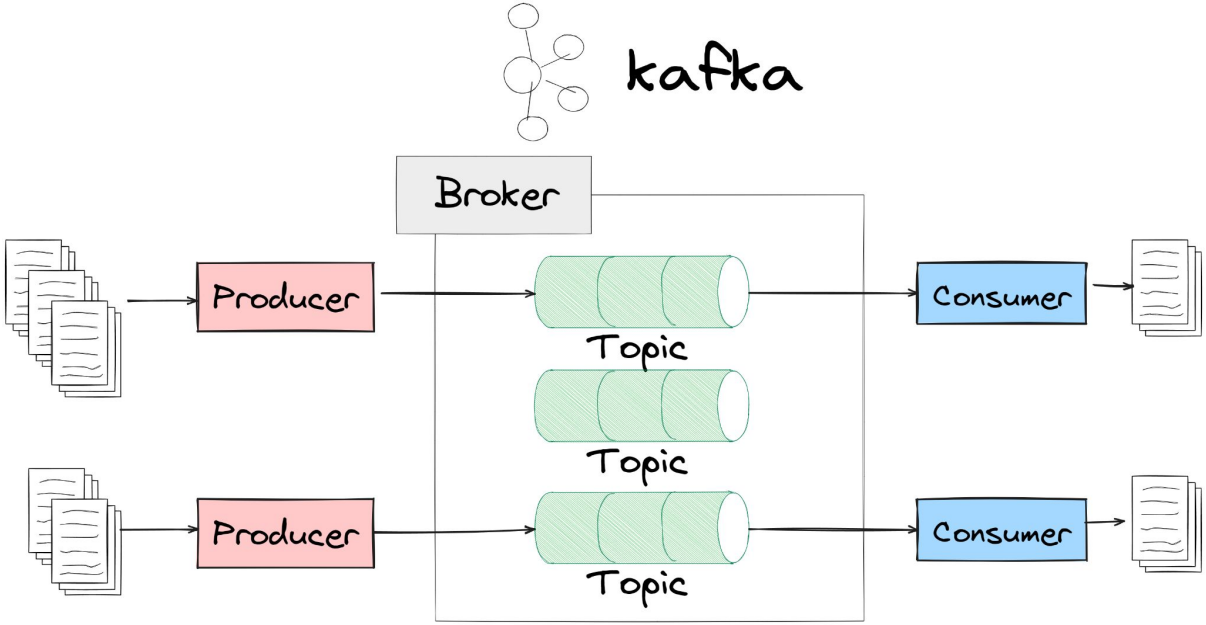
KRaft, Zookeeper

- Controller for observation, cluster control

Kafka Broker

- Represents an instance/server
- Data layer

Kafka components



Kafka examples - Usage

```
…
from kafka import KafkaProducer


producer = KafkaProducer(
 bootstrap_servers='kafka-dirk.a.aivencloud.com:13138',
 security_protocol="SSL",
 ssl_cafile="./ca.pem",
 ssl_certfile="./service.cert",
 ssl_keyfile="./service.key",
 value_serializer=lambda v: json.dumps(v).encode('ascii')
)

…

   producer.send(
    'some name',
    value=
       {
        /*…any data or messages */
       }
   )
   producer.flush()
…
```

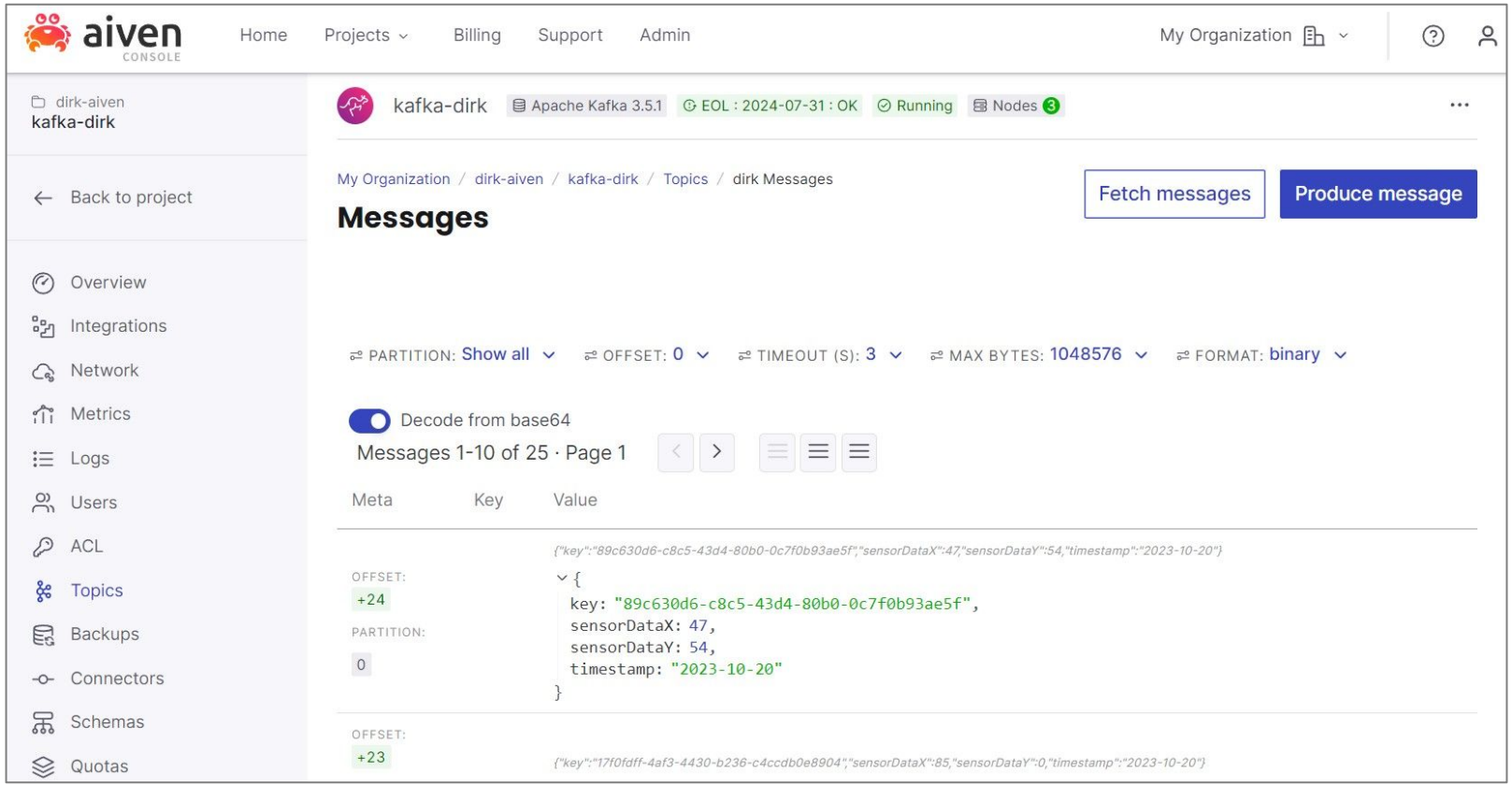
```
…
from kafka import KafkaConsumer
import json

consumer = KafkaConsumer(
 bootstrap_servers='kafka-dirk.a.aivencloud.com:13138',
 security_protocol="SSL",
 ssl_cafile="./ca.pem",
 ssl_certfile="./service.cert",
 ssl_keyfile="./service.key",
 value_deserializer = lambda v: json.loads(v.decode('ascii')),
 auto_offset_reset='earliest'
)

consumer.subscribe(topics='dirk')
for message in consumer:
  print ("%d:%d: v=%s" % (message.partition,
                          message.offset,
                          message.value))

…
```

Kafka examples - Usage



Kafka Advanced – Good 2 Know!

Message formats

- AVRO
  - recommended, less overhead
- JSON
- Protobuf

Schema Registry

- Metadata for topic content

Kafka - An easy quick start…

Installation

https://kafka.apache.org/downloads

```
# wget https://downloads.apache.org/kafka/3.7.0/kafka_2.13-3.7.0.tgz

# tar -xzf kafka_2.13-3.7.0.tgz
```

Yeah….thats it!

Kafka - An easy quick start...

Generate cluster and start Kafka with KRaft

```
# KAFKA_CLUSTER_ID="$(bin/kafka-storage.sh random-uuid)"
# bin/kafka-storage.sh format -t $KAFKA_CLUSTER_ID -c config/kraft/server.properties
# bin/kafka-server-start.sh config/kraft/server.properties
```

Or the old way.... start zookeeper and broker

```
# bin/zookeeper-server-start.sh config/zookeeper.properties
# bin/kafka-server-start.sh config/server.properties
```

Kafka - An easy quick start...

Create a topic

```
# bin/kafka-topics.sh --create --topic mytopic --bootstrap-server localhost:9092
```

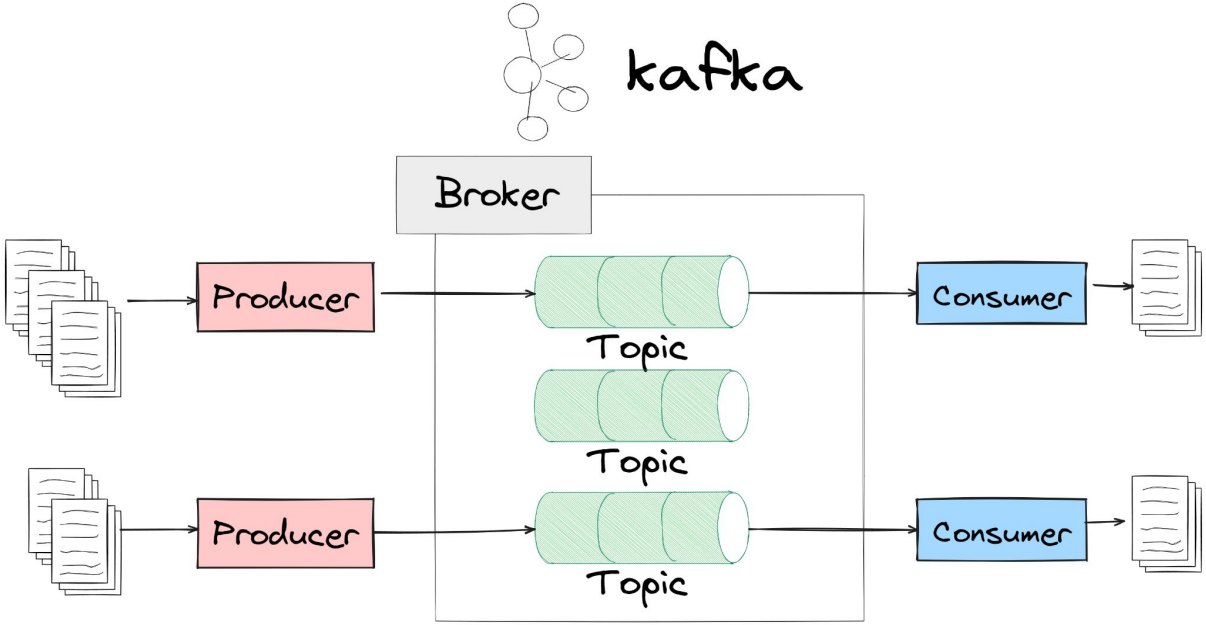
List topics

```
# bin/kafka-topics.sh --list --bootstrap-server localhost:9092
```

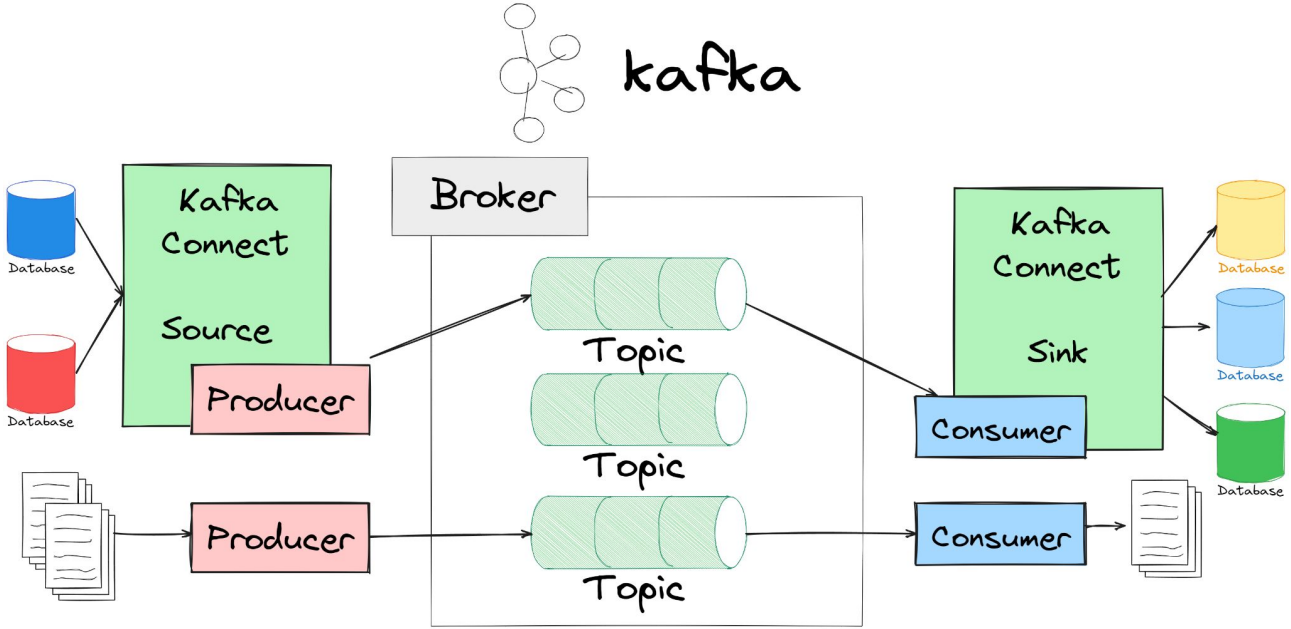
Check, produce and consume events

```
# bin/kafka-topics.sh --describe --topic mytopic --bootstrap-server localhost:9092
# bin/kafka-console-producer.sh --topic mytopic  --bootstrap-server localhost:9092
# bin/kafka-console-consumer.sh --topic mytopic  --from-beginning --bootstrap-server localhost:9092
```

Nice, but what about the databases?



Kafka Connect - Overview



Kafka Connect - Overview

Kafka Connect API

- Framework for continually pull/push
- Stand alone or distributed
- Source- and Sink-Connection

Query based approach, but ...

- ... lots of load, polling time, ordering limitations like updates/deletes

Way better... log based approach (out of the Write Ahead Log)

Let’s introduce… Debezium

Debezium

Initiated March 2016

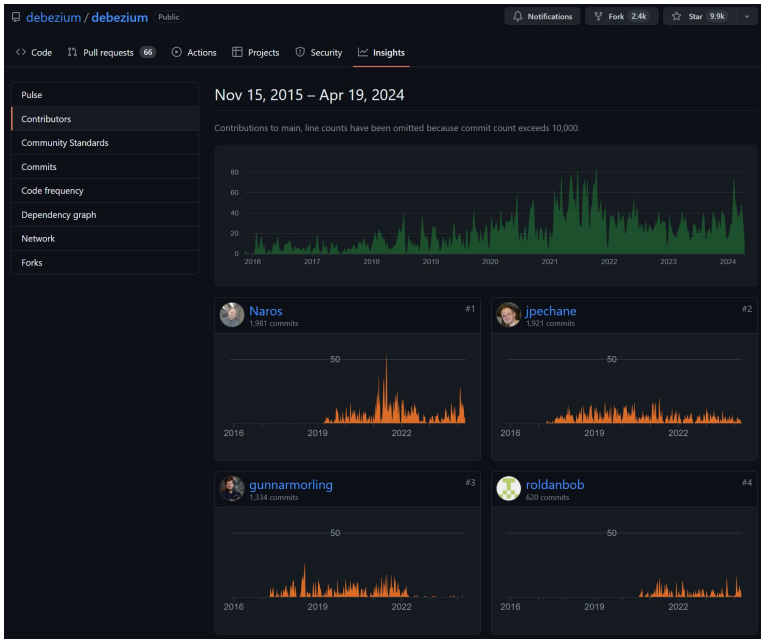
Apache License 2.0

https://github.com/debezium

Release v1.0.0 (Dec 2019)

Actual Release v2.6.2 (May 2024)

Java-based



Kafka Connect - ...with Debezium

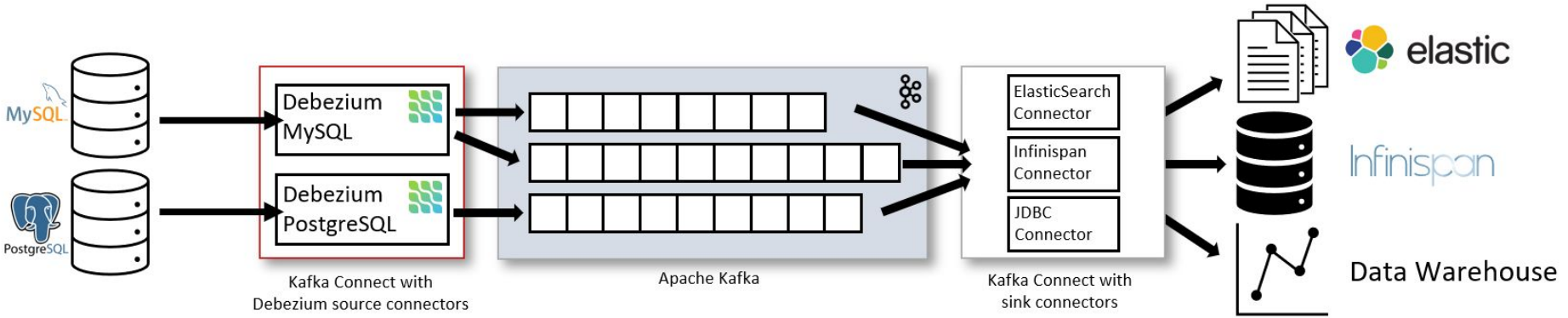
Logical Decoding API for PostgreSQL

Sink Connector for JDBC targets

REST API

Connectors for

- PostgreSQL
- Oracle
- ...



Kafka Connect - ...with Debezium

Logical replication slots

Replica Identity if no primary keys available

Publication/Subscription based

REST API

Decoding plugins

- pgoutput
- decoderbufs (additional)

Kafka Connect - A quick start...

Download and extract an connector of your choice

https://debezium.io/releases/

```
# wget https://repo1.maven.org/……/debezium-connector-postgres-2.6.2.Final-plugin.tar.gz

# mv ./debezium-connector-postgres-2.6.2.Final-plugin.tar.gz <e.g. YOUR KAFKA CLUSTER>

# cd <e.g. YOUR KAFKA CLUSTER>

# tar -xzf debezium-connector-postgres-2.6.2.Final-plugin.tar.gz
```

Kafka Connect - A quick start…

Add the extract path to e.g.

```
# vi connect-distributed.properties
```
or
```
# vi connect-standalone.properties
```

```
…
plugin.path=/home/postgres/kafka_2.13-3.7.0/
…
```

Start Kafka Connect (as standalone on same server)

```
# bin/connect-standalone.sh config/connect-standalone.properties
```

Don’t forget to prepare your database

Kafka Connect - A quick start...

Configure Debezium connector

```
curl  --location 'http://localhost:8083/connectors' \
      --header 'Accept: application/json' \
      --header 'Content-Type: application/json' \
      --data '{
   "name": "cdc-using-debezium-connector",
   "config": {
        "connector.class": "io.debezium.connector.postgresql.PostgresConnector",
        "database.hostname": "localhost",
        "database.port": "5432",
        "database.user": "postgres",
        "database.password": "postgres",
        "database.dbname": "postgres",
        "table.include.list": "public.*",
        "topic.prefix": "cdc",
        "plugin.name": "pgoutput"
   }
}'
```

Kafka Connect - Example test

```
CREATE TABLE beer (
    name varchar(255) primary key
);
ALTER TABLE beer replica identity FULL;


insert into beer values ( 'Chimay' );
insert into beer values ( 'Delirium Tremens' );
insert into beer values ( 'Leffe' );
insert into beer values ( 'Straffe Hendrik' );
insert into beer values ( 'St. Bernadus Abt 12' );

update beer set name = 'Leffe bruin' where name = 'Leffe';
update beer set name = 'Leffe royal' where name = 'Leffe bruin';
update beer set name = 'Chimay Rouge' where name = 'Chimay';

delete from beer;
```

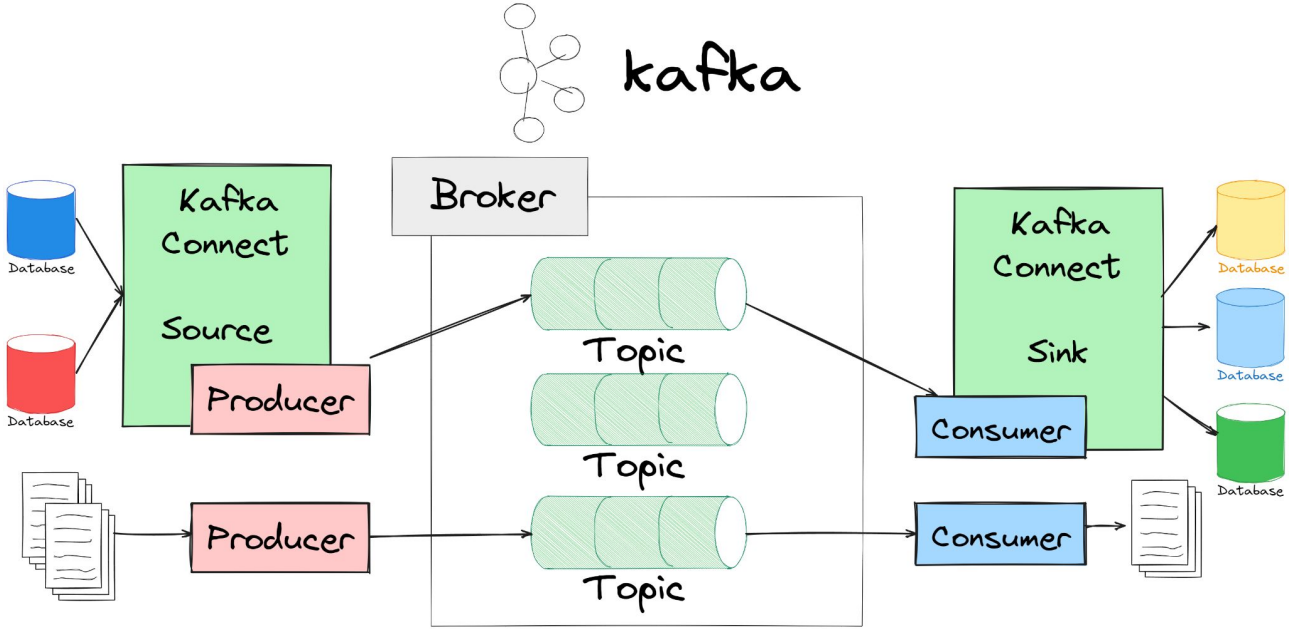
Kafka Connect - Example test

```
# bin/kafka-topics.sh --list --bootstrap-server localhost:9092
     cdc.public.artist


# bin/kafka-console-consumer.sh --topic cdc.public.artist  --from-beginning
--bootstrap-server localhost:9092

{"schema":{"type":"struct","fields":[{"type":"struct","fields":[{"type":"string",
"optional":false,"field":"name"}],"optional":true,"name":"cdc.public.artist.Value
","field":"before"},{"type":"struct","fields":[{"type":"string","optional":false,
"field":"name"}],"optional":true,"name":"cdc.public.artist.Value","field":"after"
},{"type":"struct","fields":[{"type":"string","optional":false,"field":"version"}
,{"type":"string","optional":false,"field":"connector"},{"type":"string","optiona
l":false,"field":"name"},{"type":"int64","optional":false,"field":"ts_ms"},{"type
":"string","optional":true,"name":"io.debezium.data.Enum","version":1,"parameters
":{"allowed":"true,last,...
```

Back to our nice setup....



Kafka scales pretty well, several brokers on several nodes...

But still…what if…?

Scaling vs. High Availability

Regional Outages

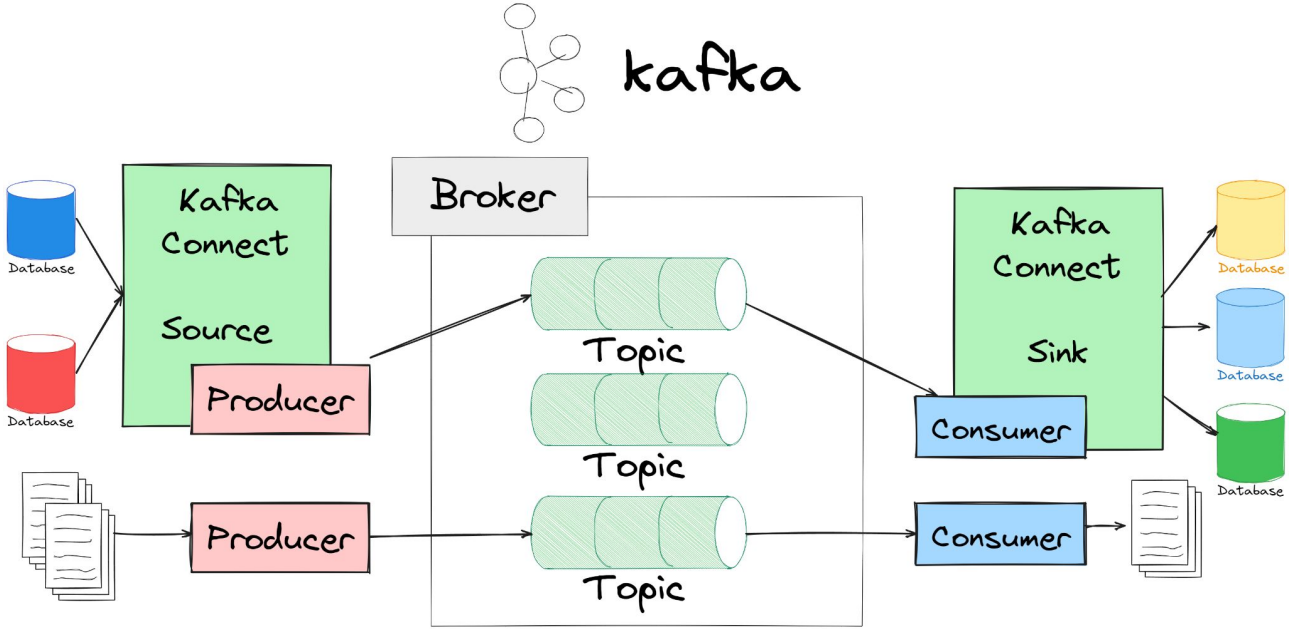
Disaster Recovery

Logical distribution

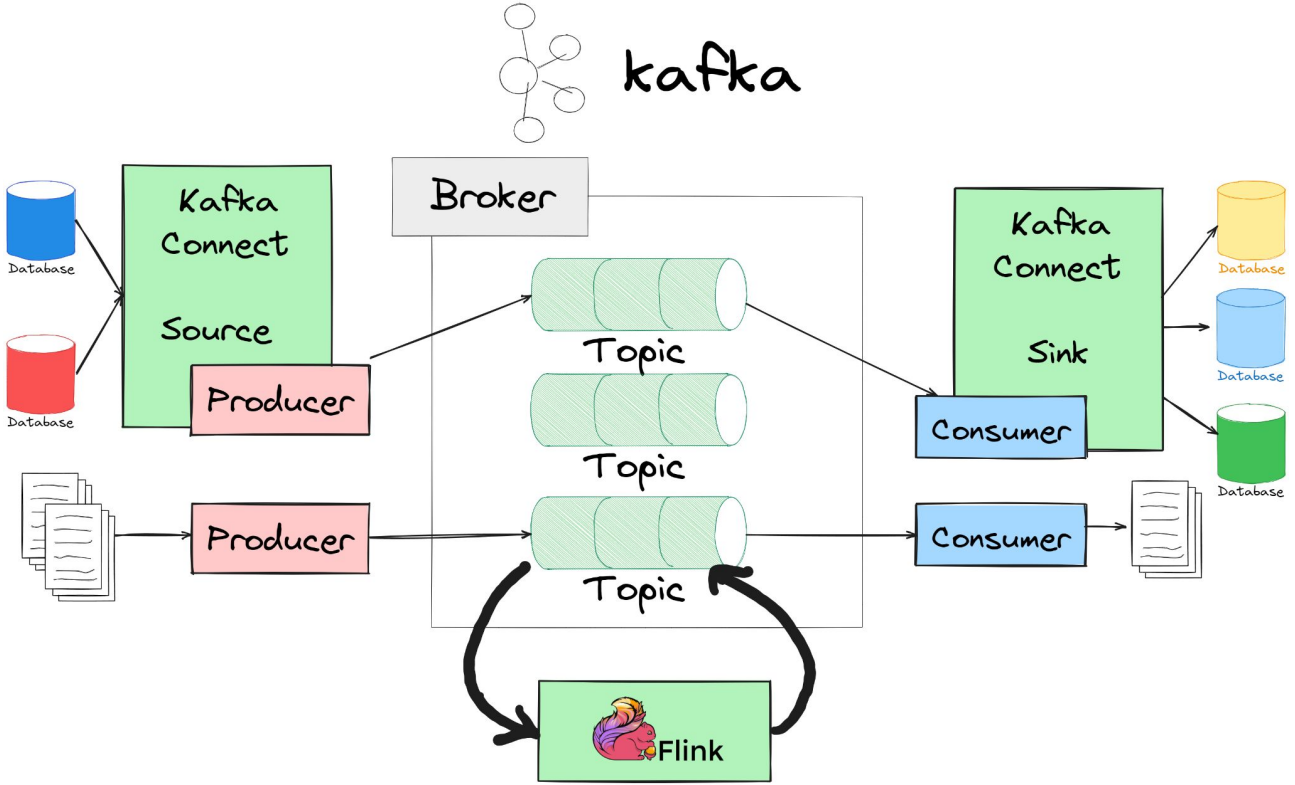
MirrorMaker 2 as a included replication tool

- Built on top of Kafka Connect framework

There is even more ...



There is even more … processing!



There is even more …

Let’s go data pipelining like a boss….!!!!

Apache Flink

- Stream processing
- Realtime
- Transforming data on the fly
- “Low latency, high throughput, high fault tolerance”

…or continue using common ETL batch ways like a caveman!

Let’s introduce… Apache Flink

Flink

Initiated Jan 2015

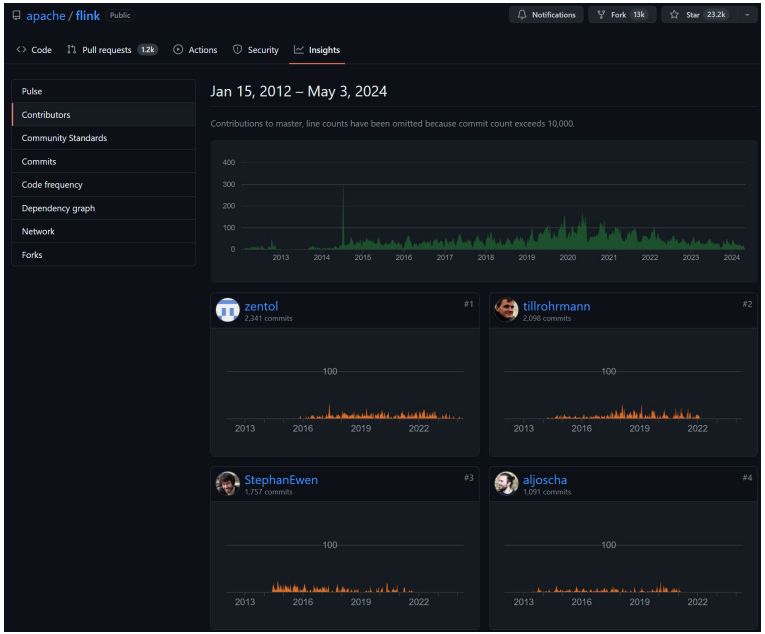
Apache License 2.0

https://github.com/apache/flink

Release v1.0.0 (March 2016)

Actual Release v1.19.0 (March 2024)

Java-, Scale-based



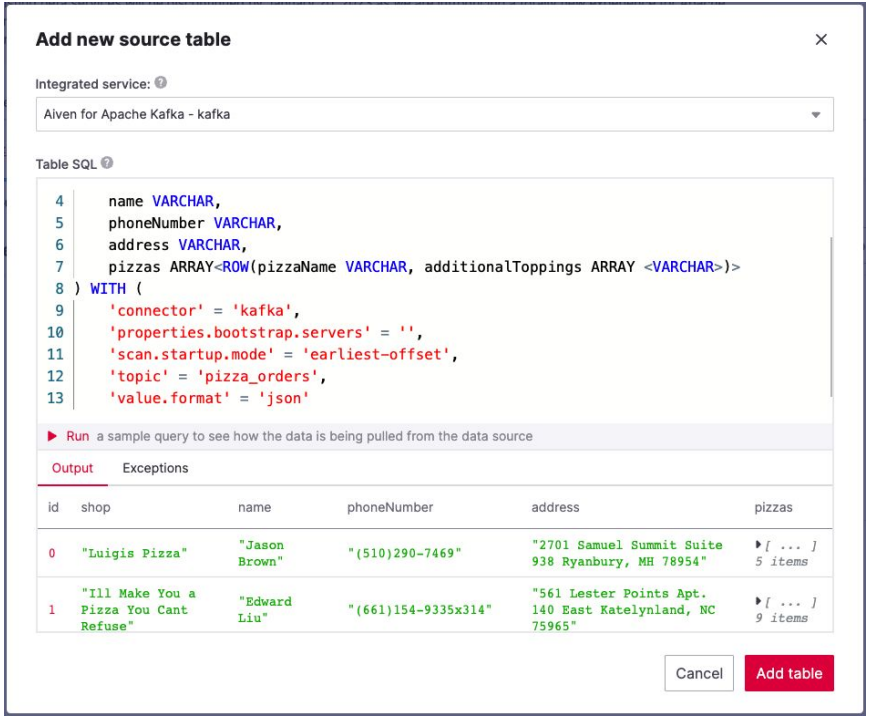
Apache Flink

“Application” as a definition of a pipeline

Creating Source- and Sink Table Definitions

Data processing using SQL

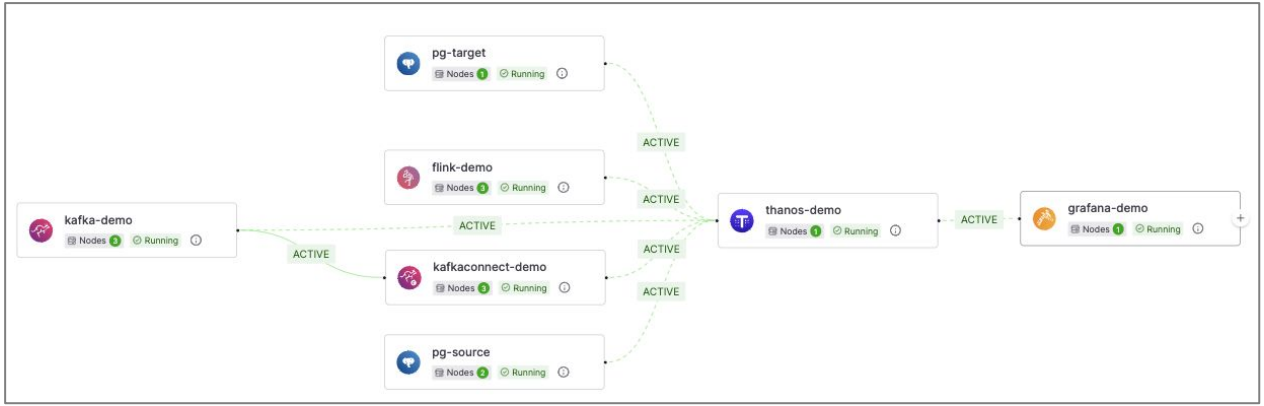
Integration of custom code with JAR file

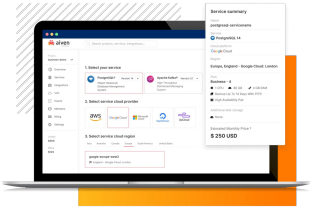


Fast, easy, reliable… -> PaaS

Easy start, challenges in real environments

Managed service recommended (no ad, but still… ;-))





Example - Be lazy using Terraform!

Create/prepare/clone from Github

https://github.com/dkrautschick/cloudland2024_pg_kafka_cdc

Adjust files (at least the var* files)

```
provider.tf
var-values.tfvars
services.tf
variables.tf
```

Run

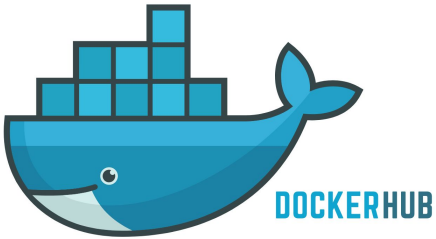
```
% terraform init
% terraform plan -var-file=var-values.tfvars
% terraform apply -var-file=var-values.tfvars -destroy
```

About Containers…

Recommended Docker Repositories

https://hub.docker.com/u/debezium

https://hub.docker.com/r/apache/kafka



Use Cases and Conclusion

Endless migration scenarios

Enhanced replication scenarios

Materialized views at source

Denormalize at target

Very powerful data pipeline stack

Flexibility, Scalability

Database migration capabilities are endless

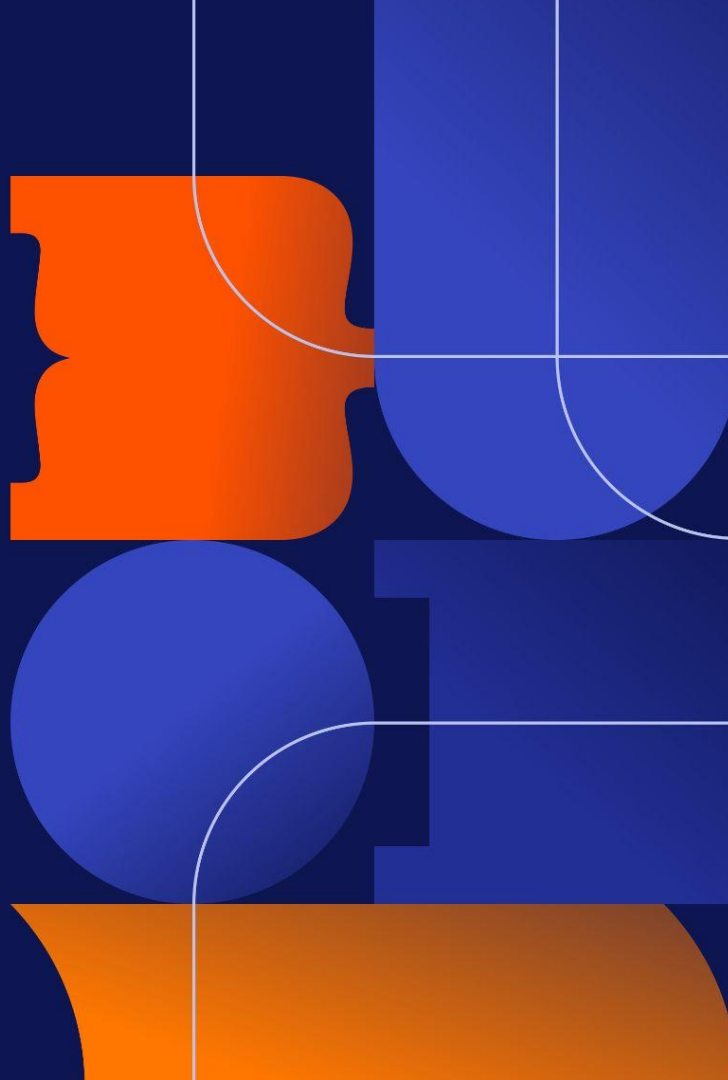
Steep learning curve

Open Source

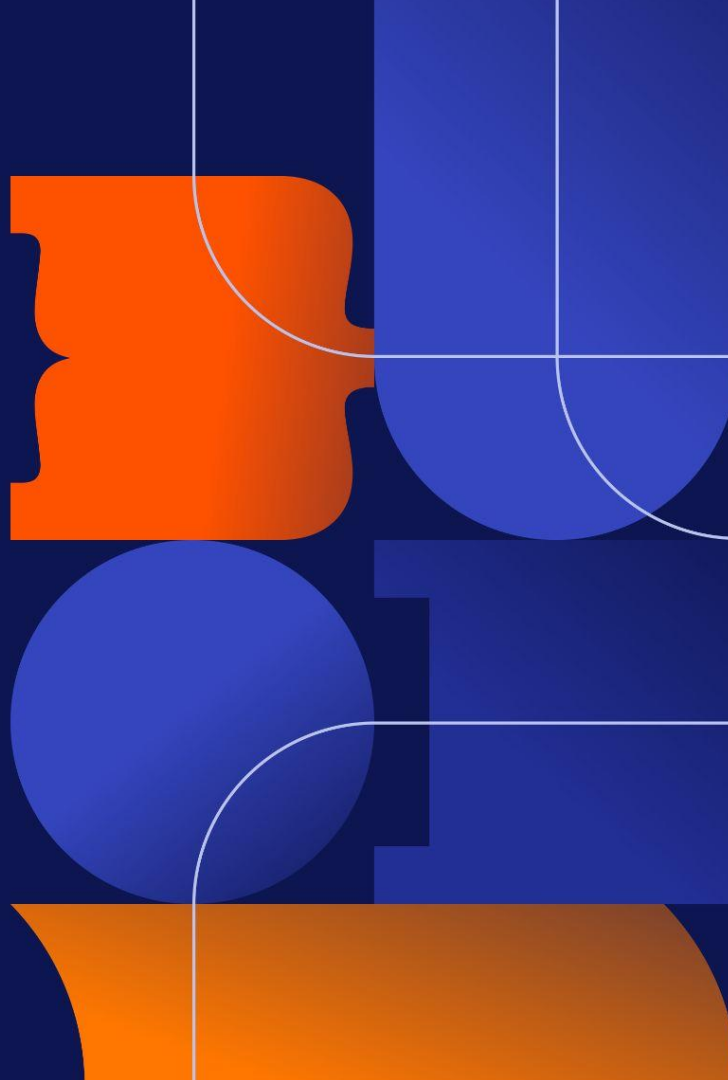
Get rid of old habits, test it yourself, use the cloud offerings!

Questions?

Please do ask or
catch me outside!



The trusted open source data platform for everyone



One data platform for your cloud needs

Event streaming	Event stream processing	Relational databases	Key-value database	Wide column database	Data warehouse	Time series database	Search engine	Data visualization
Aiven for **Apache Kafka®** and **Kafka® Connect**	Aiven for **Apache Flink®**	Aiven for **PostgreSQL®**, Aiven for **MySQL**	Aiven for **Redis®**	Aiven for **Apache Cassandra®**	Aiven for **ClickHouse®**	Aiven for **M3**	Aiven for **OpenSearch®**	Aiven for **Grafana®**

STREAM | **STORE** | **ANALYZE**

Host

aws
Google Cloud
DigitalOcean
Microsoft Azure
Bring your own cloud

Deploy

Terraform
Kubernetes
API REST API
CLI Aiven CLI
Aiven Console

Integrate

Datadog
Prometheus
AWS CloudWatch
GCP Monitoring
MongoDB
AWS S3
GCP BigQuery
Couchbase
Snowflake
splunk> Splunk
Sumologic
Debezium
GCP Pub/Sub
GCP Storage

Customers


okta


snyk


DOORDASH


priceline


fiverr.


Norauto


DECATHLON


GTL


ACTIVISION
BLIZZARD


MIRAKL


GOV.UK


goto financial


spare


Schibsted


TOYOTA


paf


CONRAD


adeo


ometria


WÄRTSILÄ